Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 54 (2007) 350–356
An International Journal
computers & mathematics with applications
www.elsevier.com/locate/camwa

A self-stabilizing algorithm for finding a minimal 2-dominating set assuming the distributed demon model

Tetz C. Huang*, Ji-Cherng Lin, Chih-Yuan Chen, Cheng-Pin Wang

Department of Computer Science and Engineering, Yuan-Ze University, 135 Yuan-Tung Road, Chung-Li, Tao-Yuan 320, Taiwan

Received 25 January 2006; accepted 22 January 2007

Abstract

A 2-dominating set in a distributed system is a set of processors such that each processor outside the set has at least two neighbors in the set. In applications, a 2-dominating set can be considered as an ideal place in the system for allocating resources, and a minimal 2-dominating set allows for the minimum of resources to be allocated. Since a maximal independent set can be viewed as a minimal 1-dominating set, the problem of finding a minimal 2-dominating set extends the problem of finding a maximal independent set in some sense. The distributed demon model for self-stabilizing systems is a natural generalization of the central demon model introduced by Dijkstra. In the past, only a few self-stabilizing algorithms under the distributed demon model have been obtained without using any transformer, and most of these algorithms are for ring networks only. In this paper, we propose a self-stabilizing algorithm that can find a minimal 2-dominating set in any general network in which the distributed demon model is assumed. This proposed algorithm is not obtained via any transformer. We also verify the correctness of the proposed algorithm.

Keywords: Self-stabilizing algorithm; Minimal 2-dominating set; Central demon model; Distributed demon model; Cut point

1. Introduction

In 1974, Dijkstra first introduced the notion of *self-stabilization* in his pioneering paper [1]. According to Dijkstra, a distributed algorithm is *self-stabilizing* if, regardless of the initial configuration of the system, any execution of the algorithm will lead the system to a legitimate configuration, and then let the system stay in the legitimate configuration (or some legitimate configurations) forever unless the system incurs a subsequent transient fault. After having been neglected for nearly a decade, Dijkstra's paper was drawn to public attention by Lamport in his invited address at PODC 1983 (cf. [2]). Since then, the research on self-stabilizing systems has flourished, and a great number of papers regarding self-stabilizing algorithms have been published. Most of these papers adopt Dijkstra's computational model, which is generally referred to as the *central demon model* (cf. [1,3,4]).

The more general *distributed demon model* (cf. [5]) was later considered by Burns [6] in 1987. The difference between the above two computational models is the number of processors that join in the execution of each atomic step of the system. Under the central demon model, exactly one among all the privileged processors in the system is

* Corresponding author.
E-mail address: cstetz@saturn.yzu.edu.tw (T.C. Huang).

0898-1221/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2007.01.021

randomly selected by the central demon to make a move in an atomic step of the system. Under the distributed demon model, however, an arbitrary number of privileged processors are randomly selected by the distributed demon to make moves simultaneously in an atomic step of the system. An atomic step of the system is also called a *system move.* Under the distributed demon model, the behavior of the system under the action of the algorithm can be described by an execution sequence $\Gamma = (\gamma_1, \gamma_2, \ldots)$ in which for any $i \geq 1$, γ_i represents a global configuration, and γ_{i+1} is obtained from γ_i after a certain number of privileged processors in the system together make the ith *system move* $\gamma_i \rightarrow \gamma_{i+1}$. The definition for an algorithm to be *self-stabilizing under the distributed demon model* is the same as under the central demon model. Thus if a system is self-stabilizing under the distributed demon model, then it is self-stabilizing under the central demon model. The converse, however, is not true (the maximal independent set algorithm in [7] is self-stabilizing under the central demon model, but not under the distributed demon model). In the past, only a few self-stabilizing algorithms under the distributed demon model have been obtained without using any transformer, and most of these algorithms are for ring networks only [6,8–10]. The reason for that is mainly due to the higher complexity of the execution of the algorithm under the distributed demon model.

A minimal 2-dominating set can be defined as follows. Suppose $G = (V, E)$ is a connected simple undirected graph. A subset D of V is a 2-*dominating set* in G if any node outside of D has at least 2 neighbors in D. A 2-dominating set D in G is *minimal* if any proper subset of D is not a 2-dominating set in G. Since a maximal independent set can be viewed as a minimal 1-dominating set, the problem of finding a minimal 2-dominating set extends the problem of finding a maximal independent set in some sense. Self-stabilizing algorithms for finding a maximal independent set have been investigated in the past. In particular, Shukla et al. have proposed in [7] a maximal independent set algorithm that is self-stabilizing under the central demon model, and Lin and Huang have proposed in [11] a fault-containing self-stabilizing algorithm for the maximal independent set problem.

In this paper, we propose a self-stabilizing algorithm that can find a minimal 2-dominating set in any general network in which the distributed demon model is assumed. This proposed algorithm is not obtained via any transformer. The result obtained in this paper extends the previous result in [7] in two directions: the minimal 2-dominating set problem extends the maximal independent set problem in some sense, and the distributed demon model is more general than the central demon model.

The rest of the paper is organized as follows. In Section 2, some graph-theoretic concepts and properties that will be needed in later discussion are either recalled or derived. In Section 3, our algorithm is presented, and the meaning of legitimate configurations is clarified. In Section 4, the correctness proof is provided, which shows that the proposed algorithm is self-stabilizing under the distributed demon model. Finally in Section 5, some remarks conclude this paper.

2. Preliminaries

In this section, some graph-theoretic concepts and properties needed in later discussion are either recalled or derived. Suppose $G = (V, E)$ is a graph. A node x in G is a *cut point of* G if the number of connected components in $G - \{x\}$ (i.e., the induced subgraph from $V - \{x\}$) is greater than that in G. If x is a cut point of G, then the degree of x in G must be at least two. For any node in G, if its degree in G is at least two and it is not a cut point of G, then we call it a *miscellaneous node of* G.

Property 1. *In a simple undirected graph, a node x is a cut point if and only if x has two distinct neighbors y and z such that the path (y, x, z) is the unique simple path in the graph that connects y and z.*

Proof. This property follows easily from [12, Theorem 1.4]. ■

Property 2. *In a simple undirected graph, a miscellaneous node must be contained in some cycle.*

Proof. Suppose x is a miscellaneous node and x is contained in a connected component C of the graph. Then x has at least two neighbors, say, y and z, and the induced subgraph $C - \{x\}$ is connected. Hence there is a simple path P in $C - \{x\}$ connecting y and z. Path P and path (y, x, z) together constitute a cycle containing x. ■

Property 3. *If x is a cut point of a connected undirected graph $G = (V, E)$, then there exist two nodes u and v such that neither u nor v is a cut point and x is on every simple path connecting u and v.*

Proof. Let $M = \{\langle y, z\rangle \mid y, z \in V$, and x is on every simple path connecting y and $z\}$ and $W = \{P \mid P$ is a simple path connecting y and z, for some $\langle y, z\rangle \in M\}$. Since x is a cut point in G, there exist two nodes y and z in G such that x is on every simple path connecting y and z (cf. [12, Theorem 1.4]). Hence $M \neq \emptyset$ and hence $W \neq \emptyset$. Let Q be the longest simple path in W and let nodes u and v be the two endpoints of Q. Then $\langle u, v\rangle \in M$ and thus x is on every simple path connecting u and v.

Suppose u or v is a cut point of G. Without loss of generality, assume that u is a cut point. Then u has two distinct neighbors l and m such that path (l, u, m) is the unique simple path that connects l and m (by Property 1). Suppose both l and m are on Q. Then the portion of Q from l to m is a simple path that connects l and m and yet does not contain u. This contradicts the fact that path (l, u, m) is the unique simple path that connects l and m. Thus either l or m is not on Q. Without loss of generality, assume that l is not on Q. Now we claim that $\langle l, v\rangle \in M$. Suppose not. Then there is a simple path P that connects l and v and yet does not contain x. Then P does not contain u (for otherwise, P contains a simple path connecting u and v and thus contains x, which causes a contradiction). Hence edge (u, l) and P together form a simple path that connects u and v, and yet does not contain x, which causes a contradiction. Therefore $\langle l, v\rangle \in M$. Let P' be the simple path formed by edge (l, u) and Q. Then $P' \in W$, and is longer than Q, which causes a contradiction. Therefore neither u nor v is a cut point. ■

Property 4. *Every nontrivial connected graph has at least two nodes that are not cut points.*

Proof. cf. [13, Theorem 3.4]. ■

3. The proposed algorithm

In this section, our self-stabilizing algorithm for solving the minimal 2-dominating set problem will be presented. The distributed system in consideration has a general underlying topology, and can be modeled by a connected simple undirected graph $G = (V, E)$, with each node $x \in V$ representing a processor in the system and each edge $\{x, y\} \in E$ representing the bidirectional link connecting processors x and y. It is assumed that

(1) each processor in the system has a unique identity,
(2) each processor x maintains two shared registers, d_x and p_x,
(3) $N(x)$ denotes the set of all neighbors of x and $L(x) = \{y \in N(x) \mid y < x\}$,
(4) the value of d_x is taken from $\{0, 1\}$,
(5) $D(x) = \{y \in N(x) \mid d_y = 1\}$, and $|D(x)|$ is the cardinality of $D(x)$, and
(6) the value of p_x is always $\emptyset$, $\{x\}$ or $D(x)$.

It should also be reiterated that the computational model assumed in the system is the distributed demon model.

Algorithm 1. {for each node x}

$R1 : d_x = 0 \wedge |D(x)| < 2 \wedge p_x = \{x\} \wedge \forall y \in \{z \in L(x) \mid d_z = 0\}, p_y \neq \{y\} \rightarrow d_x := 1$
$R2 : d_x = 1 \wedge |D(x)| \geq 2 \wedge \forall y \in N(x) - D(x), p_y = \emptyset \rightarrow d_x := 0$
$R3 : d_x = 0 \wedge |D(x)| < 2 \wedge p_x \neq \{x\} \rightarrow p_x := \{x\}$
$R4 : d_x = 0 \wedge |D(x)| = 2 \wedge p_x \neq D(x) \rightarrow p_x := D(x)$
$R5 : d_x = 0 \wedge |D(x)| > 2 \wedge p_x \neq \emptyset \rightarrow p_x := \emptyset.$

The legitimate configurations are defined to be all those configurations in which no node in the system is privileged. The following theorem clarifies that in any legitimate configuration, a minimal 2-dominating set can be identified.

Theorem 1. *If the system is in a legitimate configuration, then the set $A = \{x \in V \mid d_x = 1\}$ is a minimal 2-dominating set.*

Proof. If the system is in a legitimate configuration, then no node in the system is privileged.

(1) Suppose A is not a 2-dominating set. Then there exists a node $x \in V - A$ such that x has at most one neighbor in A, i.e., $d_x = 0$ and $|D(x)| < 2$.

Claim. *For any node u in the system, if $d_u = 0$ and $|D(u)| < 2$, then there exists a node $v \in L(u)$ (thus $v < u$) such that $d_v = 0$ and $|D(v)| < 2$.*

Proof of Claim. Since $d_u = 0$, $|D(u)| < 2$, and u is not privileged by $R3$, we have $p_u = \{u\}$. Then since u is not privileged by $R1$, $[\forall y \in \{z \in L(u) \mid d_z = 0\}, p_y \neq \{y\}]$ cannot hold. Hence there exists a node $v \in L(u)$ such that $d_v = 0$ and $p_v = \{v\}$. If $|D(v)| = 2$, then since $d_v = 0$ and $p_v = \{v\} \neq D(v)$, v is privileged by $R4$, which causes a contradiction. If $|D(v)| > 2$, then since $d_v = 0$ and $p_v = \{v\} \neq \emptyset$, v is privileged by $R5$, which causes a contradiction. Hence $|D(v)| < 2$ and the claim is proved. ■

By applying the above claim to node x, we get a $x_1 \in L(x)$ such that $d_{x_1} = 0$ and $|D(x_1)| < 2$. Then, by applying the claim to node x_1, we get a $x_2 \in L(x_1)$ such that $d_{x_2} = 0$ and $|D(x_2)| < 2$. In this way, we eventually get infinitely many nodes $x_1, x_2, x_3, \ldots$ such that $x > x_1 > x_2 > x_3 > \cdots$. However, this causes a contradiction because the system has only a finite number of nodes. Therefore, A must be a 2-dominating set.

(2) Suppose A is not a minimal 2-dominating set. Then there exists a node $x \in A$ such that $A - \{x\}$ is a 2-dominating set. Since $x \notin A - \{x\}$, x has at least two neighbors in $A - \{x\}$ and thus $|D(x)| \geq 2$. If $N(x) - D(x) = \emptyset$, then, since $d_x = 1$ and $|D(x)| \geq 2$, x is privileged by $R2$, which causes a contradiction. Hence $N(x) - D(x) \neq \emptyset$. Let y be an arbitrary node in $N(x) - D(x)$. Since $y \notin A - \{x\}$ and $A - \{x\}$ is a 2-dominating set, y has at least two neighbors, u and v, in $A - \{x\}$. Thus y has at least three neighbors u, v and x in A, i.e., $|D(y)| > 2$. Since $d_y = 0$, $|D(y)| > 2$ and y cannot be privileged by $R5$, we have $p_y = \emptyset$. Hence the condition $[\forall y \in N(x) - D(x), p_y = \emptyset]$ holds. Since $d_x = 1$, $|D(x)| \geq 2$ and $\forall y \in N(x) - D(x)$, $p_y = \emptyset$, node x is privileged by $R2$, which causes a contradiction. Hence A is a minimal 2-dominating set. ■

4. Correctness proof

In order to give a rigorous proof for the self-stabilization of Algorithm 1, we need to make the concept of execution sequence under the distributed demon model more precise. A sequence $\Gamma = (\gamma_1, \gamma_2, \ldots)$ is called an *infinite execution* (of Algorithm 1 under the distributed demon model) if

(a) Γ is an infinite sequence,
(b) for any $m = 1, 2, \ldots, \gamma_m$ is a global configuration,
(c) for any $m = 1, 2, \ldots, \gamma_{m+1}$ is induced from γ_m after a certain number of privileged processors selected by the distributed demon make the system move $\gamma_m \to \gamma_{m+1}$.

A sequence $\Gamma = (\gamma_1, \gamma_2, \ldots)$ is called a *finite execution* (of Algorithm 1 under the distributed demon model) if

(a) Γ is a finite sequence and $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_q)$ for some $q \in Z^+$,
(b) for any $m = 1, 2, \ldots, q$, γ_m is a global configuration,
(c) for any $m = 1, 2, \ldots, q-1$, γ_{m+1} is induced from γ_m after a certain number of privileged processors selected by the distributed demon make the system move $\gamma_m \to \gamma_{m+1}$,
(d) no node in the system is privileged in the last configuration γ_q.

Thus, unless clearly specified, an execution $\Gamma = (\gamma_1, \gamma_2, \ldots)$ may refer to an infinite or a finite execution. The following lemma is quite obvious in view of the definition of a legitimate configuration, the definition of a finite execution and the definition of an algorithm being self-stabilizing under the distributed demon model.

Lemma 1. *Algorithm 1 is self-stabilizing under the distributed demon model if and only if any execution of Algorithm 1 is a finite execution.*

For presentation's sake, we define some more notation and terminology. For any configuration γ of $G = (V, E)$, let $V_\gamma = \{x \in V \mid d_x = 1 \text{ in } \gamma\}$ and let $G_\gamma = (V_\gamma, E_\gamma)$ be the induced subgraph of G from V_γ, i.e., $E_\gamma = E \cap \{\{x, y\} \mid x, y \in V_\gamma\}$. Suppose $\gamma \to \gamma'$ is a system move. We say that

(1) a connected component C is *divided due to the system move* $\gamma \to \gamma'$, if C is a connected component of G_γ and there exist two distinct nodes x and y in C such that x and y are in different connected components of $G_{\gamma'}$,
(2) a connected component C *completely disappears due to the system move* $\gamma \to \gamma'$, if C is a connected component of G_γ and no node in C remains in $G_{\gamma'}$, and
(3) a cycle H *disappears due to the system move* $\gamma \to \gamma'$, if H is a cycle in G_γ and at least one node in H does not remain in $G_{\gamma'}$.

Lemma 2. *Any two adjacent nodes in G cannot both execute rule $R1$ in a same system move.*

Proof. Suppose x and y, $x < y$, are two adjacent nodes that execute $R1$ in a same system move $\gamma \to \gamma'$. Then $d_x = 0$ and $p_x = \{x\}$ in γ. Hence the condition $[\forall u \in \{z \in L(y) \mid d_z = 0\}, p_u \neq \{u\}]$ for y cannot hold in γ, and thus y cannot execute $R1$ in $\gamma \to \gamma'$. A contradiction occurs. Hence the lemma is proved. ■

Lemma 3. *For any system move $\gamma \to \gamma'$, if P is a simple path in $G_{\gamma'}$ and its two endpoints are also in G_γ, then the path P is also in G_γ.*

Proof. Let x and y be the two endpoints of P. Suppose P is not in G_γ. Then there exists a node z in P such that z is not in G_γ. Thus $z \neq x$ and $z \neq y$ (i.e., z is not an endpoint of P), and z must execute $R1$ in the system move $\gamma \to \gamma'$. If the two neighbors of z in P are in G_γ, then $|D(z)| \geq 2$ in γ and z is not privileged by $R1$ in γ. A contradiction occurs. Hence a neighbor u of z in P is not in G_γ and u must execute $R1$ in the system move $\gamma \to \gamma'$. Thus two adjacent nodes z and u both execute $R1$ in the system move $\gamma \to \gamma'$, which contradicts Lemma 2. Therefore, P must be in G_γ. ■

Lemma 4. *For any system move $\gamma \to \gamma'$, if H is a cycle in $G_{\gamma'}$, then H is also a cycle in G_γ. (That is, no new cycle can result from any system move.)*

Proof. Suppose H is not a cycle in G_γ. Then H contains a node x not in G_γ. Since x is in $G_{\gamma'}$, x must execute $R1$ in the system move $\gamma \to \gamma'$. Since $G_{\gamma'}$ is simple, cycle H must pass through at least three nodes. Hence x has two neighbors in H. For the rest of the proof, one can argue analogously as in the proof of the preceding lemma. ■

Lemma 5. *For any system move $\gamma \to \gamma'$, if a connected component C completely disappears due to $\gamma \to \gamma'$, then C contains a miscellaneous node in G_γ (and thus the miscellaneous node executes $R2$ in the system move $\gamma \to \gamma'$).*

Proof. Since no node in C remains in $G_{\gamma'}$, every node in C executes $R2$ in the system move $\gamma \to \gamma'$. Thus for every node x in C, $|D(x)| \geq 2$. Hence C contains at least three nodes and every node in C is either a cut point or a miscellaneous node. Since C is a nontrivial connected component, by Property 4, C contains at least two nodes that are not cut points, and thus are miscellaneous nodes. ■

Lemma 6. *If there exists a node executing $R2$ in the system move $\gamma \to \gamma'$, then either a cycle disappears due to $\gamma \to \gamma'$ or a connected component is divided due to $\gamma \to \gamma'$.*

Proof. If a node x executes $R2$ in the system move $\gamma \to \gamma'$, then since $d_x = 1$ and $|D(x)| \geq 2$ in γ, $x \in G_\gamma$ and $\deg(x) \geq 2$ in G_γ. Thus x is either a cut point or a miscellaneous node in G_γ.

Case 1. There is a miscellaneous node x in G_γ executing $R2$ in $\gamma \to \gamma'$. Then x is contained in a cycle H in G_γ (by Property 2). Since x executes $R2$ in $\gamma \to \gamma'$ and changes d_x to 0, x is not in $G_{\gamma'}$. This implies that cycle H disappears due to $\gamma \to \gamma'$. So we are done in this case.

Case 2. There is no miscellaneous node in G_γ executing $R2$ in $\gamma \to \gamma'$. Then only cut points can possibly execute $R2$ in $\gamma \to \gamma'$. This, together with the condition of the lemma, implies that there is a cut point x in G_γ executing $R2$ in $\gamma \to \gamma'$. Let C be the connected component of G_γ that contains x. By Property 3, there exist two nodes u and v in C such that neither u nor v is a cut point and x is on every simple path in C connecting u and v. Since neither u nor v is a cut point in C, neither u nor v can execute $R2$ in $\gamma \to \gamma'$. Thus u and v are in $G_{\gamma'}$. Suppose there exists a simple path P in $G_{\gamma'}$ connecting u and v. By Lemma 3, P is also in G_γ. Then, since P is a simple path in C connecting u and v, x is in P and thus also in $G_{\gamma'}$. This causes a contradiction (since x executes $R2$ in $\gamma \to \gamma'$ and changes d_x to 0 in γ', x cannot be in $G_{\gamma'}$). Hence there is no simple path in $G_{\gamma'}$ connecting u and v, i.e., nodes u and v are in different connected components of $G_{\gamma'}$. Therefore connected component C is divided due to $\gamma \to \gamma'$. So we are done in this case. ■

Theorem 2. *Algorithm 1 is self-stabilizing under the distributed demon model.*

Proof. Suppose not. Then, in view of Lemma 1, there must exist an infinite execution Γ of Algorithm 1 under the distributed demon model.

Claim 1. *Γ contains infinitely many system moves in each of which at least one node in the system executes $R2$.*

Proof of Claim 1. Suppose Γ contains only a finite number of system moves in each of which at least one node in the system executes $R2$. Then there exists a suffix Γ^* of Γ such that no node in the system G executes $R2$ in Γ^*. Hence in Γ^*, each node in the system can execute $R1$ at most once. Therefore, there exists a suffix Γ^{**} of Γ^* such that in Γ^{**}, no node executes $R1$ or $R2$. Thus no node changes its d-value in Γ^{**} and hence each node can execute $R3$, $R4$ or $R5$ at most once in Γ^{**}. It follows that Γ is a finite execution, which causes a contradiction. Hence the claim is proved. ■

Let $\Gamma = (\gamma_1, \gamma_2, \ldots)$. Since the number of cycles in G_{γ_1} is finite and no new cycle results from any system move $\gamma_i \to \gamma_{i+1}$ in Γ (by Lemma 4), there are only a finite number of system moves $\gamma_i \to \gamma_{i+1}$ in Γ such that some cycle disappears due to $\gamma_i \to \gamma_{i+1}$. Therefore, there exists a suffix Γ^* of Γ such that

(A) no cycle disappears due to any system move in Γ^*.

If a node in the system executes $R2$ in a system move $\gamma \to \gamma'$, then, by Lemma 6, either a cycle disappears due to $\gamma \to \gamma'$ or a connected component is divided due to $\gamma \to \gamma'$. Therefore, in view of (A), we have that

(B) if a node in the system executes $R2$ in a system move $\gamma \to \gamma'$ in Γ^*, then at least one connected component is divided due to $\gamma \to \gamma'$.

Since each miscellaneous node is contained in a cycle (by Property 2) and no cycle disappears due to any system move in Γ^* (by (A) above), no miscellaneous node can execute $R2$ in any system move in Γ^*. Thus, by Lemma 5, we have that

(C) no connected component completely disappears due to any system move in Γ^*.

Let $\Gamma^* = (\gamma_q, \gamma_{q+1}, \ldots)$ and for any configuration γ_i in Γ^*, let α_{γ_i} be the number of connected components of G_{γ_i}. Then for any system move $\gamma_i \to \gamma_{i+1}$ in Γ^*, no connected component completely disappears due to $\gamma_i \to \gamma_{i+1}$ (by (C) above) and any two nodes in two different components of G_{γ_i} cannot be in the same connected component in $G_{\gamma_{i+1}}$ (by Lemma 2). Therefore, we have that

(D) for any system move $\gamma_i \to \gamma_{i+1}$ in Γ^*, $\alpha_{\gamma_i} \leq \alpha_{\gamma_{i+1}}$.

If, furthermore, a node in the system executes $R2$ in a system move $\gamma_i \to \gamma_{i+1}$, then at least a connected component is divided due to $\gamma_i \to \gamma_{i+1}$. This, together with the two observations before (D), implies that

(E) if a node in the system executes $R2$ in a system move $\gamma_i \to \gamma_{i+1}$ in Γ^*, then $\alpha_{\gamma_i} < \alpha_{\gamma_{i+1}}$.

Since Γ^* contains infinitely many system moves in each of which at least one node in the system executes $R2$ (by Claim 1 above), this, together with (D) and (E) above, implies that $\lim_{i\to\infty} \alpha_{\gamma_i} = \infty$. However, since $\alpha_{\gamma_i} \leq n$ for any γ_i in Γ^* (where n is the number of nodes in the system), $\lim_{i\to\infty} \alpha_{\gamma_i} \not> n$. So we have a contradiction here and therefore the theorem is proved. ■

5. Concluding remarks

In the above, we have proposed a self-stabilizing algorithm for finding a minimal 2-dominating set in a general network in which the distributed demon model is assumed. We have also provided the correctness proof for the self-stabilization of the proposed algorithm. Our result extends the previous result in [7].

As mentioned previously, only a few self-stabilizing algorithms under the distributed demon model have been obtained without using any transformer, and most of these algorithms are for ring networks only. Moreover, there exists a counterexample showing that an existing self-stabilizing algorithm under the central demon model may not be self-stabilizing under the distributed demon model. In [5], it is pointed out that even if a self-stabilizing algorithm under the central demon model is actually self-stabilizing under the distributed demon model, it may not be easy to verify the correctness. In this research, we actually started with a version of algorithm that is similar to the self-stabilizing maximal independent set algorithm in [7]. That version is self-stabilizing under the central demon model and solves the minimal 2-dominating set problem in a general network. However, it is not self-stabilizing under the distributed demon model. We therefore had to search for an algorithm that works. The algorithm in this paper actually

results from that earlier version after several rounds of modification, and whether it works or not was really uncertain until the correctness proof was acquired.

Finally, we would like to mention that a natural direction for further investigation along this research line is to design a self-stabilizing algorithm for finding a minimal k-dominating set (k is an arbitrary positive integer) in a general network that assumes the distributed demon model.

References

[1] E.W. Dijkstra, Self-stabilizing systems in spite of distributed control, Commun. ACM 17 (1974) 643–644.
[2] L. Lamport, Solved problems, unsolved problems and non-problems in concurrency, in: PODC 1984 Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing, 1984, pp. 1–11. Invited address.
[3] E.W. Dijkstra, Self-stabilization in spite of distributed control, in: Selected Writings on Computing: A Personal Perspective, Springer-Verlag, Berlin, 1982, pp. 41–46.
[4] E.W. Dijkstra, A belated proof of self-stabilization, Distrib. Comput. 1 (1986) 5–6.
[5] T.C. Huang, A self-stabilizing algorithm for the shortest path problem assuming the distributed demon, Comput. Math. Appl. 50 (2005) 671–681.
[6] J.E. Burns, Self-stabilizing ring without demons, Technical Report GIT-ICS-87/36, Georgia Tech., 1987.
[7] S. Shukla, D.J. Rosenkrantz, S.S. Ravi, Observations on self-stabilizing graph algorithms on anonymous networks, in: Proceedings of the 2nd Workshop on Self-Stabilizing Systems, WSS, LasVegas, Nevada, 1995, pp. 7.1–7.15.
[8] J.E. Burns, M.G. Gouda, R.E. Miller, On relaxing interleaving assumptions, in: Proceedings of the MCC Workshop on Self-Stabilizing Systems, MCC Technical Report STP-379-89, Austin, Texas, 1989.
[9] G.M. Brown, M.G. Gouda, C.L. Wu, Token systems that self-stabilize, IEEE Trans. Comput. 38 (1989) 845–852.
[10] M.S. Tsai, S.T. Huang, Self-stabilizing ring orientation protocols, in: Proceedings of the Second Workshop on Self-Stabilizing Systems, WSS, Las Vegas, Nevada, 1995, pp. 16.1–16.14.
[11] J.C. Lin, T.C. Huang, An efficient fault-containing self-stabilizing algorithm for finding a maximal independent set, IEEE Trans. Parallel Distrib. Syst. 14 (2003) 742–754.
[12] F. Buckley, F. Harary, Distance in Graphs, Addison-Wesley Publishing Company, California, 1990.
[13] F. Harary, Graph Theory, Addison-Wesley Publishing Company, Massachusetts, 1969.